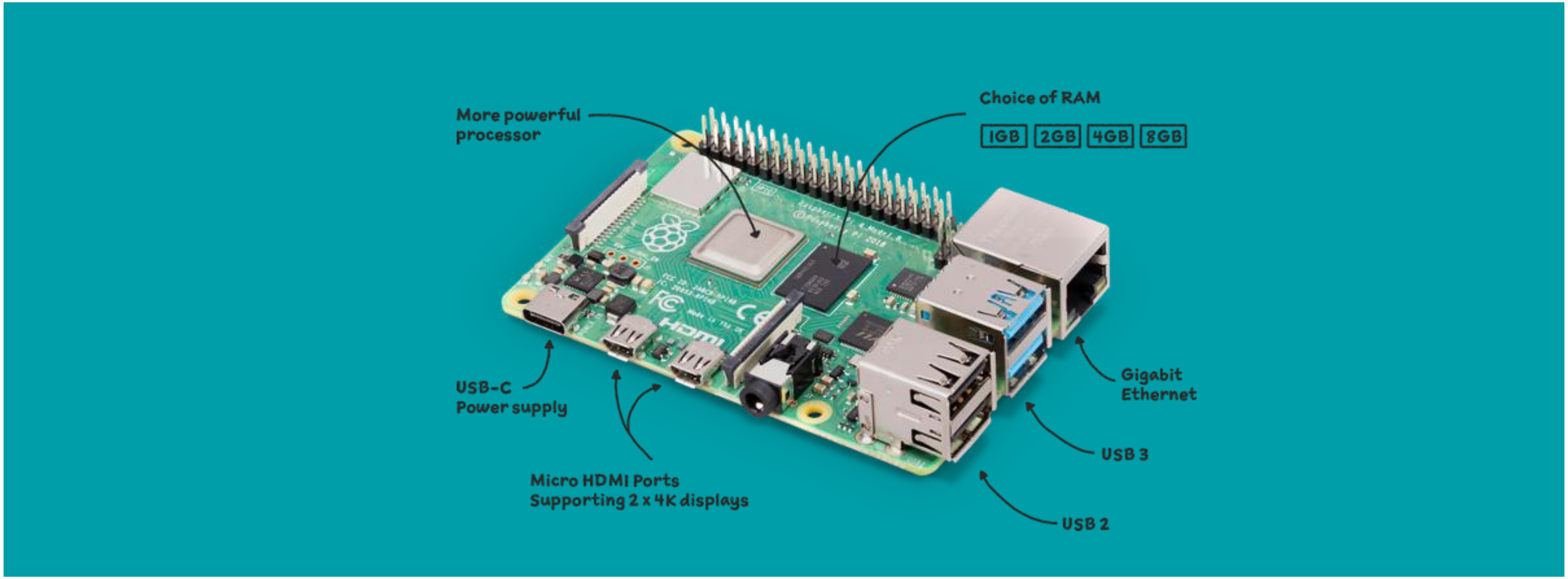


6. 센서활용-4

이영주
young.kopo@gmail.com

한국폴리텍대학

Class 모듈 만들기(통신, I2C)

i2c_class.py 파일

```
import smbus
import math

class read_i2c:
    bus = None
    i2c_address = None
    command = 0x44
    def __init__(self):
        self.bus = smbus.SMBus(1)
        ……
    def  vr_read():
        …..
        return VrValue
   ……..
class write_i2c:
    state = 0b00000000
    bus = None
```

◆ read_i2c 관련 메서드
- vr_read()
  - 반환 값: 백분율의 소수점 2자리까지
- cds_read()
  - 반환 값: 백분율의 소수점 2자리까지
- gas_read()
  - 반환 값: 0~ 1023 정수값(AD검출값)
- psd_read()
  - 반환 값: 거리값(cm), 소수점 2자리까지

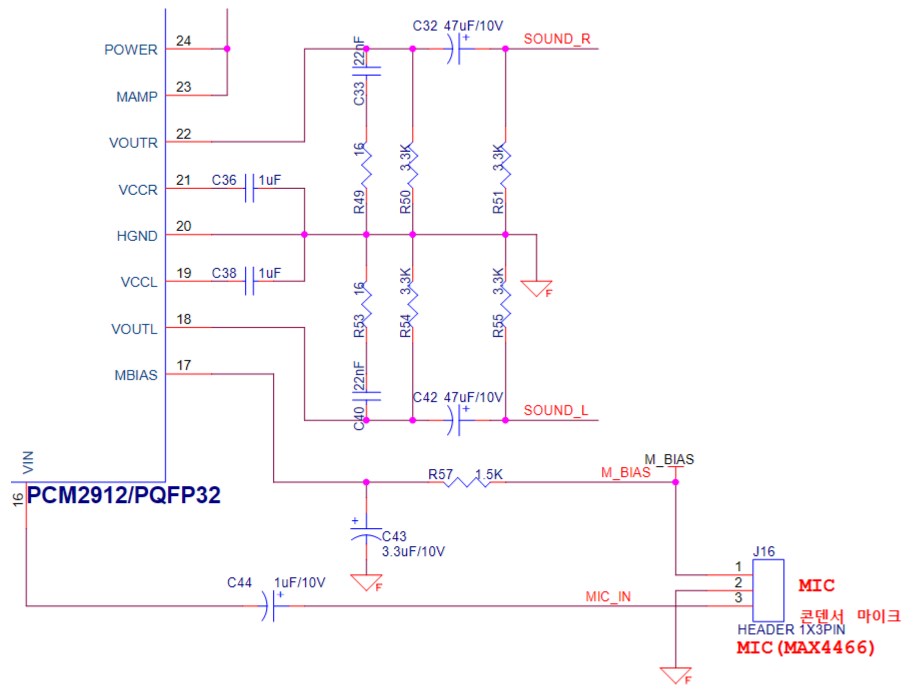
◆ write_i2c 관련 메서드
- On(cmd)
  - cmd: 제어비트 On
  - 반환 값: 없음
- Off(cmd)
  - cmd: 제어비트 Off
  - 반환 값: 없음

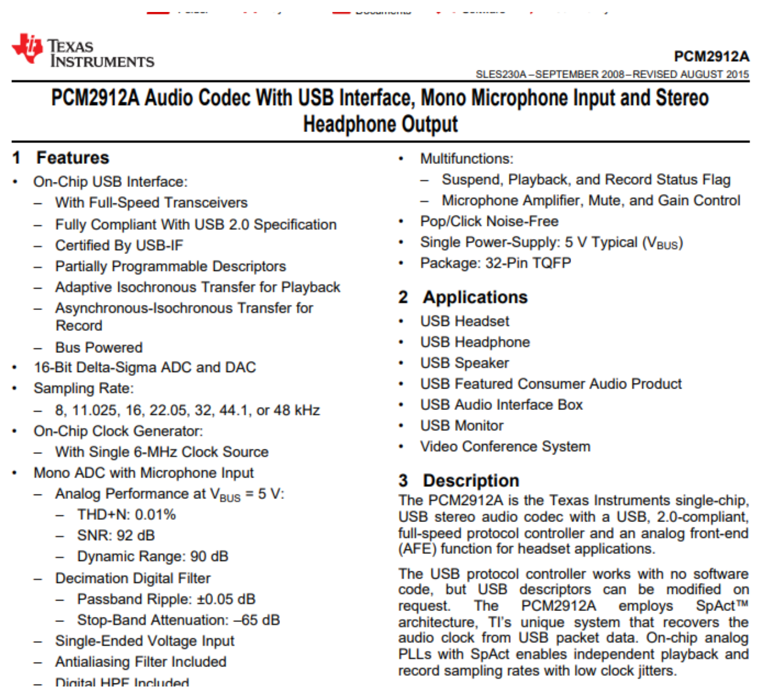
Class 모듈 만들기(통신, I2C)

```
from i2c_class import read_i2c
from i2c_class import write_i2c
import I2C_LCD_driver
import time

RED_LED    = 0b00000001
GREEN_LED = 0b00000010
BLUE_LED   = 0b00000100
RELAY_1    = 0b00010000
RELAY_2    = 0b00100000

textLcd = I2C_LCD_driver.lcd()
textLcd.lcd_display_string("KOPO AISW", 1)
textLcd.lcd_display_string("I2C_BUS TEST", 2)

adc = read_i2c()
write = write_i2c()

print("VR:"+str(adc.vr_read()) +" %")
print("CdS:"+str(adc.cds_read()) + " %")
print("GAS:"+str(adc.gas_read())+ " GAS")
print("Distance:"+str(adc.psd_read())+ " cm")
textLcd.lcd_display_string(str(adc.vr_read())+" %" + str(adc.cds_read()) + " %", 2)
```

```
write.On(RED_LED)
time.sleep(0.5)
write.On(GREEN_LED)
time.sleep(0.5)
write.On(BLUE_LED)
time.sleep(0.5)
write.Off(RED_LED| GREEN_LED| BLUE_LED)
time.sleep(0.5)
write.On(RELAY_1 | RELAY_2)
time.sleep(0.5)
write.Off(RELAY_1 | RELAY_2)
time.sleep(0.5)
```

스피커 제어(통신, I2C)




TEXAS INSTRUMENTS

PCM2912A

SLES230A – SEPTEMBER 2008 – REVISED AUGUST 2015

PCM2912A Audio Codec With USB Interface, Mono Microphone Input and Stereo Headphone Output

1 Features

- On-Chip USB Interface:
  - With Full-Speed Transceivers
  - Fully Compliant With USB 2.0 Specification
  - Certified By USB-IF
  - Partially Programmable Descriptors
  - Adaptive Isochronous Transfer for Playback
  - Asynchronous-Isochronous Transfer for Record
  - Bus Powered
- 16-Bit Delta-Sigma ADC and DAC
- Sampling Rate:
  - 8, 11.025, 16, 22.05, 32, 44.1, or 48 kHz
- On-Chip Clock Generator:
  - With Single 6-MHz Clock Source
- Mono ADC with Microphone Input
  - Analog Performance at V_{BUS} = 5 V:
    - THD+N: 0.01%
    - SNR: 92 dB
    - Dynamic Range: 90 dB
  - Decimation Digital Filter
    - Passband Ripple: ±0.05 dB
    - Stop-Band Attenuation: –65 dB
  - Single-Ended Voltage Input
  - Antialiasing Filter Included
  - Digital HPF Included
- Multifunctions:
  - Suspend, Playback, and Record Status Flag
  - Microphone Amplifier, Mute, and Gain Control
- Pop/Click Noise-Free
- Single Power-Supply: 5 V Typical (V_{BUS})
- Package: 32-Pin TQFP

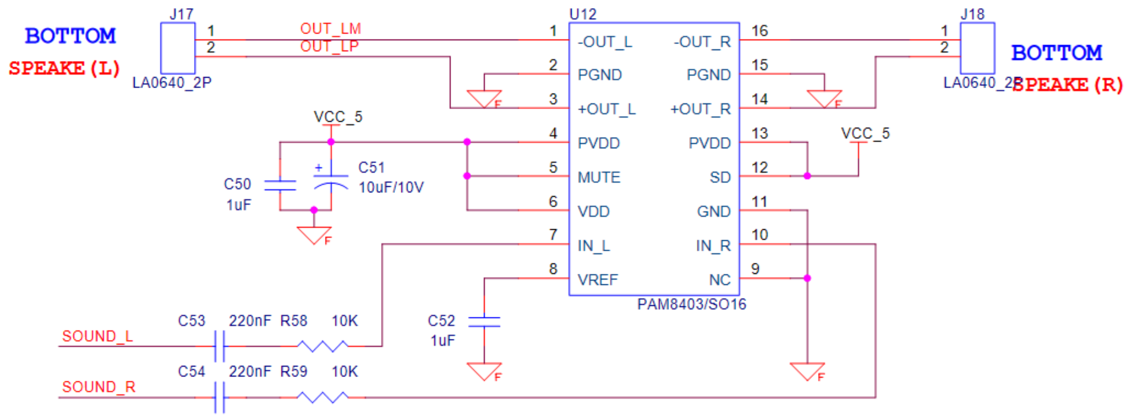
2 Applications

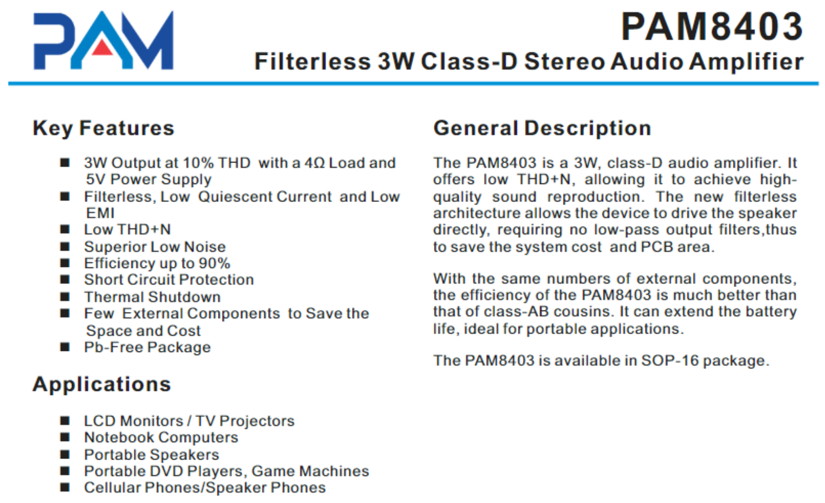
- USB Headset
- USB Headphone
- USB Speaker
- USB Featured Consumer Audio Product
- USB Audio Interface Box
- USB Monitor
- Video Conference System

3 Description

The PCM2912A is the Texas Instruments single-chip, USB stereo audio codec with a USB, 2.0-compliant, full-speed protocol controller and an analog front-end (AFE) function for headset applications.

The USB protocol controller works with no software code, but USB descriptors can be modified on request. The PCM2912A employs SpAct™ architecture, TI's unique system that recovers the audio clock from USB packet data. On-chip analog PLLs with SpAct enables independent playback and record sampling rates with low clock jitters.




PAM

PAM8403

Filterless 3W Class-D Stereo Audio Amplifier

Key Features

- 3W Output at 10% THD with a 4Ω Load and 5V Power Supply
- Filterless, Low Quiescent Current and Low EMI
- Low THD+N
- Superior Low Noise
- Efficiency up to 90%
- Short Circuit Protection
- Thermal Shutdown
- Few External Components to Save the Space and Cost
- Pb-Free Package

Applications

- LCD Monitors / TV Projectors
- Notebook Computers
- Portable Speakers
- Portable DVD Players, Game Machines
- Cellular Phones/Speaker Phones

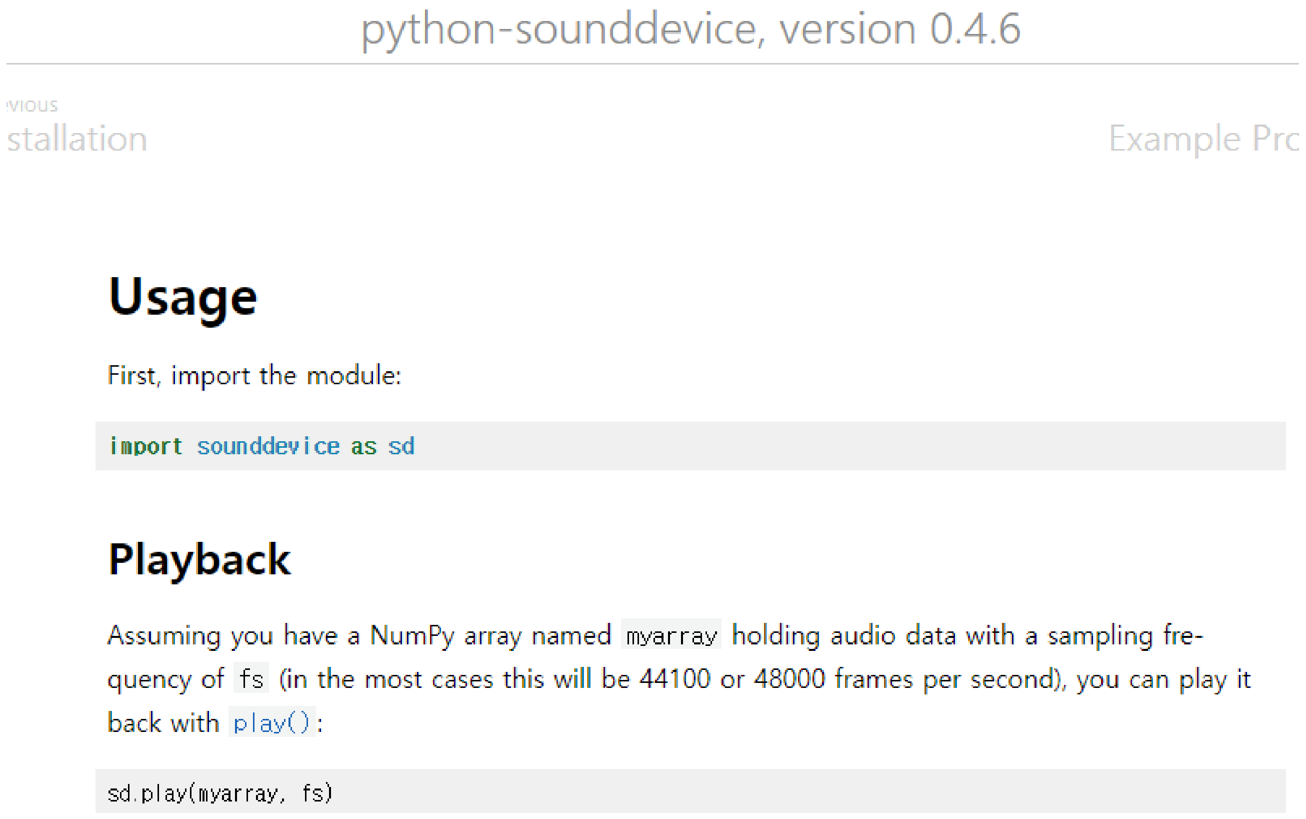
General Description

The PAM8403 is a 3W, class-D audio amplifier. It offers low THD+N, allowing it to achieve high-quality sound reproduction. The new filterless architecture allows the device to drive the speaker directly, requiring no low-pass output filters,thus to save the system cost and PCB area.

With the same numbers of external components, the efficiency of the PAM8403 is much better than that of class-AB cousins. It can extend the battery life, ideal for portable applications.

The PAM8403 is available in SOP-16 package.

스피커 제어(통신, I2C)



python-sounddevice, version 0.4.6

vious
stallation

Example Prc

Usage

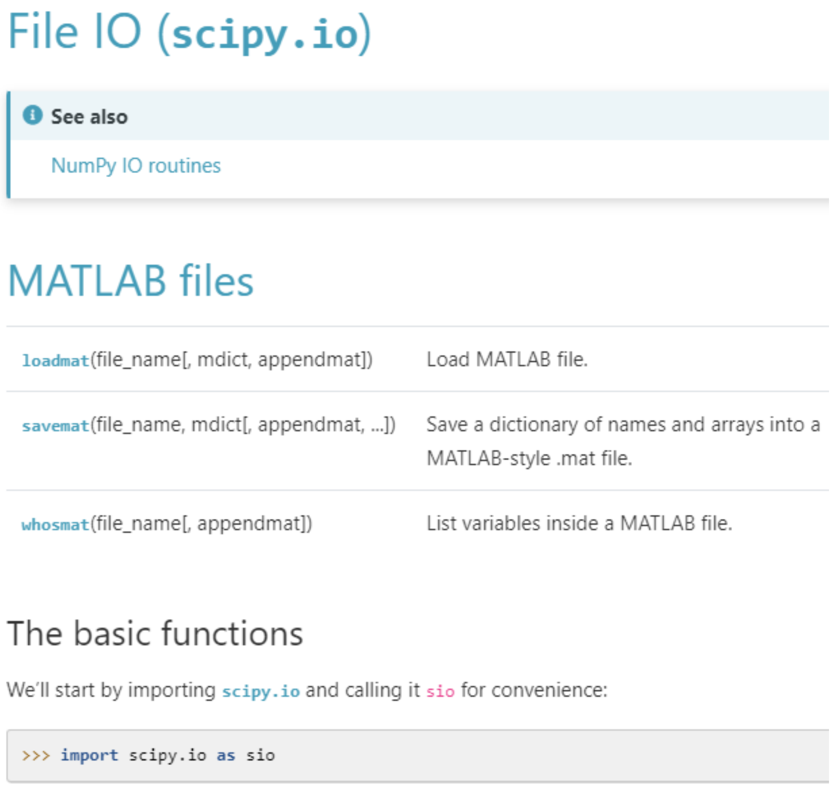
First, import the module:

```
import sounddevice as sd
```

Playback

Assuming you have a NumPy array named `myarray` holding audio data with a sampling frequency of `fs` (in the most cases this will be 44100 or 48000 frames per second), you can play it back with `play()`:

```
sd.play(myarray, fs)
```



File IO (`scipy.io`)

See also

NumPy IO routines

MATLAB files

`loadmat`(file_name[, mdict, appendmat])	Load MATLAB file.
`savemat`(file_name, mdict[, appendmat, ...])	Save a dictionary of names and arrays into a MATLAB-style .mat file.
`whosmat`(file_name[, appendmat])	List variables inside a MATLAB file.

The basic functions

We'll start by importing `scipy.io` and calling it `sio` for convenience:

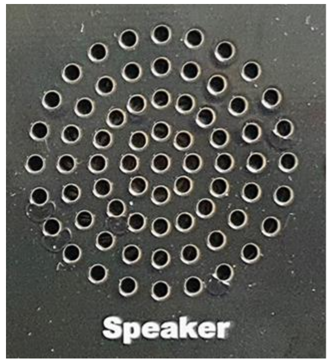
```
>>> import scipy.io as sio
```

마이크 입력 및 저장(통신, I2C)

```
import sounddevice as sd
import scipy.io as sio
import scipy.io.wavfile

sample_rate = 44100  # 샘플레이트
seconds = 3  # 녹음시간

myrecording = sd.rec(int(seconds * sample_rate), samplerate=sample_rate, channels=2)
sd.wait()  # 녹음이 끝날때까지 대기
rint(“Recording Stop”)
sio.wavfile.write('output.wav', sample_rate, myrecording)  # wav파일로 저장
```

마이크 및 스피커 제어(통신, I2C)

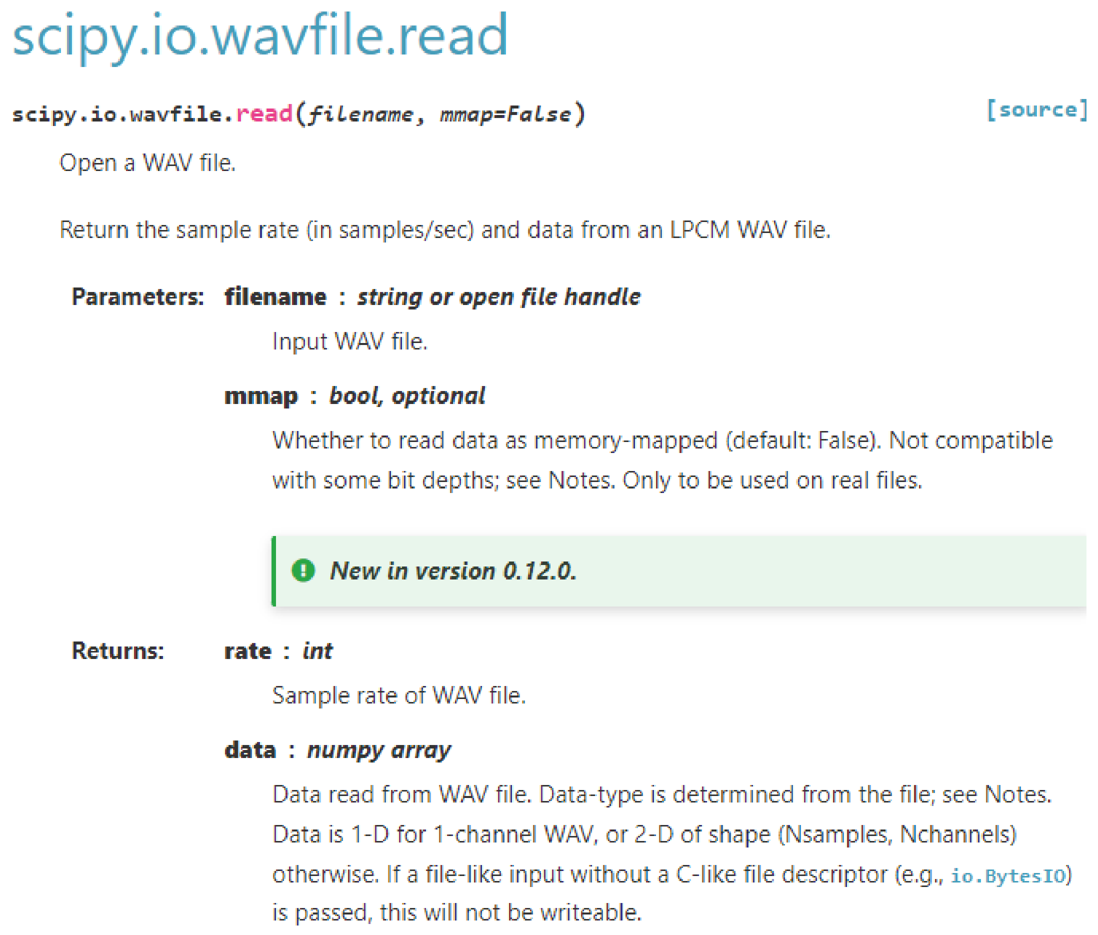
```
import pygame
import time

pygame.mixer.init()
p = pygame.mixer.Sound('output.wav')
p.play()
time.sleep(3)
```



Wave file 제어(통신, I2C)

재생 시간 측정

```
import numpy as np
import scipy.io as sio
import scipy.io.wavfile

fileNamePath = 'output.wav'
samplerate, data = sio.wavfile.read(fileNamePath)
times =  data.shape[0]/samplerate
print("file '{}' play time = {}".format(fileNamePath, times))
```



scipy.io.wavfile.read

`scipy.io.wavfile.read(filename, mmap=False)` [source]

Open a WAV file.

Return the sample rate (in samples/sec) and data from an LPCM WAV file.

Parameters: **filename** : ***string or open file handle***

Input WAV file.

mmap : ***bool, optional***

Whether to read data as memory-mapped (default: False). Not compatible with some bit depths; see Notes. Only to be used on real files.

New in version 0.12.0.

Returns: **rate** : ***int***

Sample rate of WAV file.

data : ***numpy array***

Data read from WAV file. Data-type is determined from the file; see Notes. Data is 1-D for 1-channel WAV, or 2-D of shape (Nsamples, Nchannels) otherwise. If a file-like input without a C-like file descriptor (e.g., `io.BytesIO`) is passed, this will not be writeable.

Wave file 제어_재생시간 측정 및 재생(통신, I2C)

```
import numpy as np
import scipy.io as sio
import scipy.io.wavfile
import pygame
import time

fileNamePath = 'output.wav'
# 샘플레이트 및 데이터를 통해 재생 시간을 구한다.
samplerate, data = sio.wavfile.read(fileNamePath)
times = data.shape[0]/samplerate

# 반올림 및 형변환을 통해 정수형태로 구한다.
play_time = int(round(times))
print("file '{}' play time = {}".format(fileNamePath, play_time ))
# 재생을 위한 pygame 초기화
pygame.mixer.init()

# 경로를 지정하여 객체를 생성하고 재생한다.
p = pygame.mixer.Sound(fileNamePath)
p.play()

# 위에서 구한 파일의 길이 만큼 대기한다.
time.sleep(play_time)
```

Wave file 제어_재생시간 측정 및 재생(통신, I2C)

```
import sounddevice as sd
import numpy as np
import scipy.io as sio
import scipy.io.wavfile
import pygame
import time

fileNamePath = 'output.wav'

sample_rate = 44100  # 샘플레이트
seconds = 3  # 녹음시간
print("{}초 동안 녹음을 시작합니다.".format(seconds))

myrecording = sd.rec(int(seconds * sample_rate), samplerate=sample_rate, channels=2)
sd.wait()  # 녹음이 끝날때까지 대기
sio.wavfile.write(fileNamePath, sample_rate, myrecording)  # wav파일로 저장

# 샘플레이트 및 데이터를 통해 재생 시간을 구한다.
samplerate, data = sio.wavfile.read(fileNamePath)
times = data.shape[0]/samplerate

# 반올림 및 형변환을 통해 정수형태로 구한다.
play_time = int(round(times))

# 재생을 위한 pygame 초기화
pygame.mixer.init()

# 경로를 지정하여 객체를 생성하고 재생한다.
p = pygame.mixer.Sound(fileNamePath)
p.play()

# 위에서 구한 파일의 길이 만큼 대기한다.
time.sleep(play_time)
```